Grado en Ingeniería Informática
Grado en Matemáticas e Informática

Asignatura: PROGRAMACIÓN II

Pruebas-1: prueba de programas

Profesores Programación II

DLSIIS - E.T.S. de Ingenieros Informáticos
Universidad Politécnica de Madrid

Febrero 2015

Probar programas

- Para comprobar que un programa funciona como esperamos, lo que hemos hecho hasta ahora es usar un método **main** para crear objetos, llamar a métodos sobre esos objetos e imprimir por consola.
- Otra manera muy habitual de probar un programa es usando la herramienta **junit**

El *framework* JUnit

- JUnit es una librería y marco de trabajo (*framework*) que permite definir y automatizar pruebas de unidades (clases y métodos).
- Facilita el proceso de repetir las pruebas.
- Está desarrollado para Java.
- Hay versiones para otros lenguajes.
- Se puede encontrar en http://www.junit.org
- La versión actual es la 4.11.

JUnitTest

- Supongamos que queremos comprobar que un método f de una clase A que acabamos de escribir es correcto.
- Podríamos escribir un **JUnitTest** para f
- Un JUnitTest es una clase que contiene una serie de métodos test.
- Cada método test para f debe realizar N veces lo siguiente:
  - Llamar a f con unos ciertos argumentos
  - Comprobar que f produce el resultado correcto.
- Los métodos test pueden realizar otras comprobaciones que se estudiarán más adelante.

Ejecución de un JUnitTest en Eclipse

- Se debe incluir dentro del mismo proyecto eclipse en el que se encuentra el código que queremos probar.
- El proyecto debe importar la librería *junit-4.11.jar* o utilizar la librería interna de Eclipse.
- Se debe ejecutar como “JUnit Test”.
- El orden de ejecución de los métodos test dentro del JunitTest es indeterminado.
- Eclipse genera un informe en el que se muestra el resultado de la ejecución de cada test.

Definición de un método test

- JUnit trabaja con anotaciones '@'
  - Una **anotación** es como una etiqueta que proporciona información adicional acerca del elemento (método) que viene a continuación.
- @Test: Indica que el método que viene a continuación es un método test (prueba):

```
/**
 * Test method for {@link Almacen#test0Constructor()}.
 */
@Test
public void test0Constructor() {
   …
}
```

Preparación de las pruebas

- A veces antes de ejecutar los métodos test, es necesario ejecutar otros métodos para realizar los preparativos de las pruebas.
  - Para eso JUnit proporciona los métodos *BeforeClass* y *Before* que se especifican mediante las siguientes anotaciones:
    - **@BeforeClass**: Sólo se realiza una vez antes de que se ejecute la primera prueba
    - **@Before**: Se ejecuta justo antes de cada prueba

Ejemplo test junit

```
public class TestAlmacen {
  private Almacen almacen;


@Before
public void setUp() {
  almacen = new Almacen(3);
}

/**
 * Test method for {@link Almacen#test0Constructor()}.
 */
@Test
public void test0Constructor() {
  int[] salida = {0,0,0};
  assertArrayEquals(salida, almacen.getProductos());
}
```

Definición de un método test

- Para comprobar si los resultados producidos por un método que se está probando son correctos, vamos a utilizar aserciones.
- Para implementar una aserción se pueden utilizar los siguientes métodos definidos en el *framework* JUnit:
  - **static void assertTrue**(boolean condition)
  - **static void assertTrue**(String msg, boolean condition)
  - **static void assertEquals**(String msg, Object expected, Object actual)
  - **static void assertArrayEquals**(Object[] expected, Object[] actuals)
  - etc.

Ejecución de un método test

- Después de ejecutar un test, éste se marca en el informe de resultados como:
  - **Error:** Se ha producido una excepción no esperada o ha vencido un timeout
  - **Failure**: No se ha cumplido un assert o no se ha generado una excepción esperada

Definición de un método test

- ¿Cómo detectamos que no tarda mucho?
  - **@Test** (timeout = milisegundos)
- ¿Cómo probamos que un método genera una excepción cuando debe?
  - **@Test** (expected = exception.class)

Finalización de las pruebas

- Igual que hemos visto las etiquetas *BeforeClass* y *Before* para los métodos que se usan para preparar las pruebas, también hay unas etiquetas equivalentes para la finalización:
  - **@AfterClass**: se ejecuta al terminar todas las pruebas
  - **@After**: se ejecuta después de cada prueba